CSC 261/461
Database Systems

Eustrat Zhupa

December 4, 2018

Structured Query Language

Transactions

- Updates triggered by real-world events: receipt of a new order from a customer.
  1. Add new order to table of orders.
  2. Update sales total for the salesperson who took the order.
  3. Update sales total for the salesperson's office.
  4. Update the available total for the ordered product.

Structured Query Language

Transactions

- A sequence of one or more SQL statements that together form a logical unit of work.
  - typically closely related and perform interdependent actions.
  - each statement in the transaction performs some part of a task
  - all of them are required to complete the task.
- Grouping the statements as a single transaction means sequence should be executed *atomically*.

Transactions

Add-an-order

To accept a customer's order, the order-entry program should

1. query the PRODUCTS table to ensure that the product is in stock,
2. insert the order into the ORDERS table
3. update the PRODUCTS table, subtracting the quantity ordered from the quantity-on-hand of the product,
4. update the SALESREPS table, adding the order amount to the total sales of the salesperson who took the order
5. update the OFFICES table, adding the order amount to the total sales of the office where the salesperson works.

Transactions

Cancel-an-order

To cancel a customers order, the program should

1. delete the order from the ORDERS table
2. update the PRODUCTS table, adjusting the quantity-on-hand total for the product,
3. update the SALESREPS table, subtracting the order amount from the salespersons total sales
4. update the OFFICES table, subtracting the order amount from the office's total sales.

Transactions

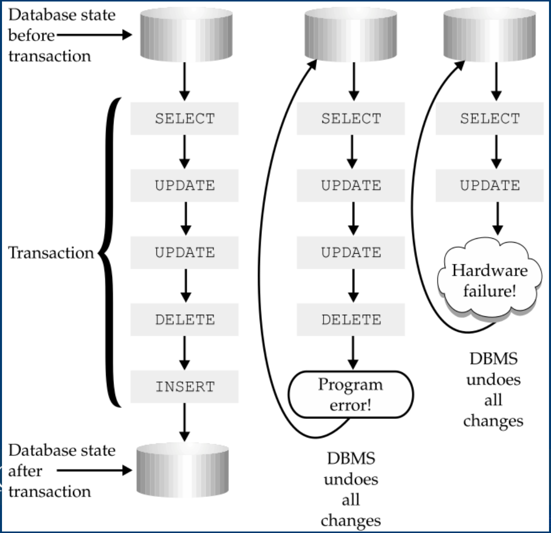
Reassign-a-customer

When a customer is reassigned from one salesperson to another, the program should

1. update the CUSTOMERS table to reflect the change
2. update the ORDERS table to show the new salesperson for all orders placed by the customer,
3. update the SALESREPS table, reducing the quota for the salesperson losing the customer, and
4. update the SALESREPS table, raising the quota for the salesperson gaining the customer

Transaction Concept

The statements in a transaction will be executed as an atomic unit of work in the database. Either all of the statements will be executed successfully, or none of the statements will be executed.



Structured Query Language

COMMIT and ROLLBACK

- SQL supports database transactions through
  1. `COMMIT` signals the successful end of a transaction.
     - tells the DBMS transaction is complete
     - all of the statements in the transaction have been executed
     - the database is consistent.
  2. `ROLLBACK` signals unsuccessful end of a transaction.
     - tells the DBMS the user does not want to complete the transaction
     - DBMS should back out any changes made to the database during the transaction.
     - DBMS restores the database to its state before the transaction began.

Transactions

Transaction Log

- When an SQL statement modifies the database, the DBMS automatically writes a record in the transaction log
  1. One copy shows the row before the change
  2. the other copy shows the row after the change
- when log is written DBMS modifies row on disk.
- If user subsequently executes a COMMIT statement, the end-of-transaction is noted in the transaction log.
- If user executes a ROLLBACK, the DBMS restores the rows to earlier state.

Transactions

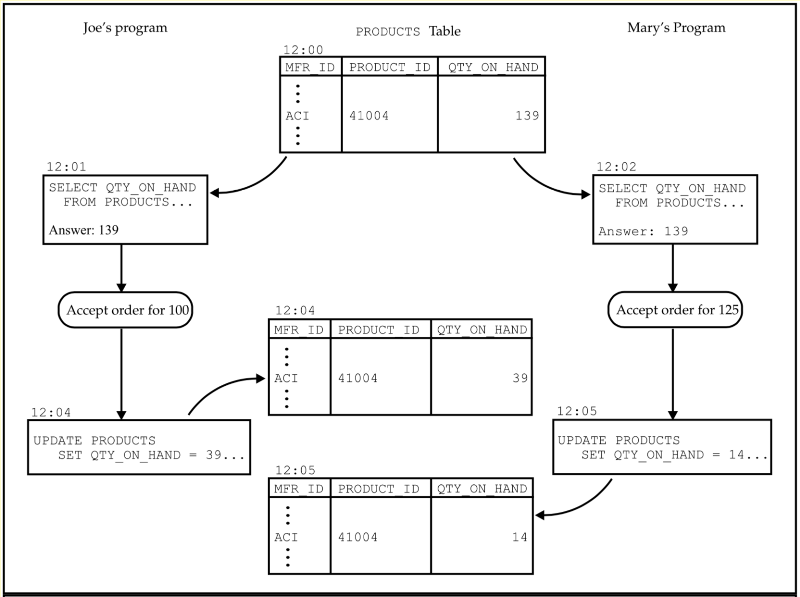
Multiuser Transactions

When two or more users concurrently access a database

- recover properly from system failures or errors
- ensure that the users' actions do not interfere with one another.
- each user accesses the database as if he or she had exclusive access to it, without worrying about the actions of other users.

Structured Query Language

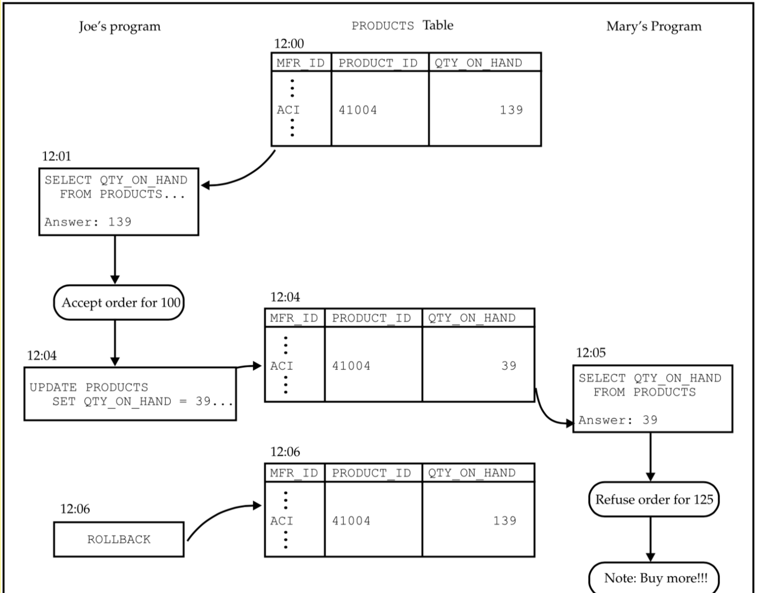
Lost Update



Joe's program
PRODUCTS Table
Mary's Program
12:00
MFR_ID | PRODUCT_ID | QTY_ON_HAND
ACI | 41004 | 139
12:01
SELECT QTY_ON_HAND FROM PRODUCTS...
Answer: 139
12:02
SELECT QTY_ON_HAND FROM PRODUCTS...
Answer: 139
Accept order for 100
Accept order for 125
12:04
MFR_ID | PRODUCT_ID | QTY_ON_HAND
ACI | 41004 | 39
12:04
UPDATE PRODUCTS SET QTY_ON_HAND = 39...
12:05
UPDATE PRODUCTS SET QTY_ON_HAND = 14...
12:05
MFR_ID | PRODUCT_ID | QTY_ON_HAND
ACI | 41004 | 14

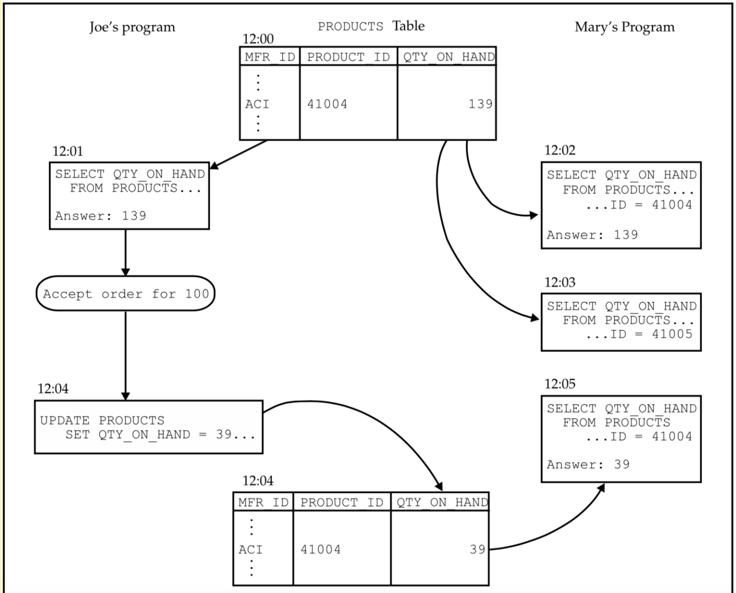
Structured Query Language

Dirty Read (*P*1)



Structured Query Language

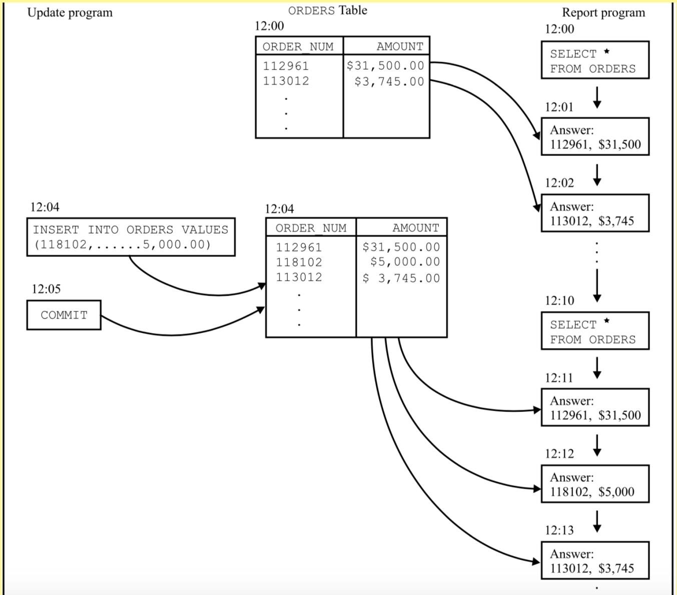
Nonrepeatable Read (*P*2)



Joe's program
PRODUCTS Table
Mary's Program
12:00
MFR_ID
PRODUCT_ID
QTY_ON_HAND
ACI
41004
139
12:01
SELECT QTY_ON_HAND FROM PRODUCTS...
Answer: 139
12:02
SELECT QTY_ON_HAND FROM PRODUCTS... ...ID = 41004
Answer: 139
Accept order for 100
12:03
SELECT QTY_ON_HAND FROM PRODUCTS... ...ID = 41005
12:04
UPDATE PRODUCTS SET QTY_ON_HAND = 39...
12:05
SELECT QTY_ON_HAND FROM PRODUCTS ...ID = 41004
Answer: 39
12:04
MFR_ID
PRODUCT_ID
QTY_ON_HAND
ACI
41004
39

Structured Query Language

Phantom Insert ($P3$)



Concurrent Transactions

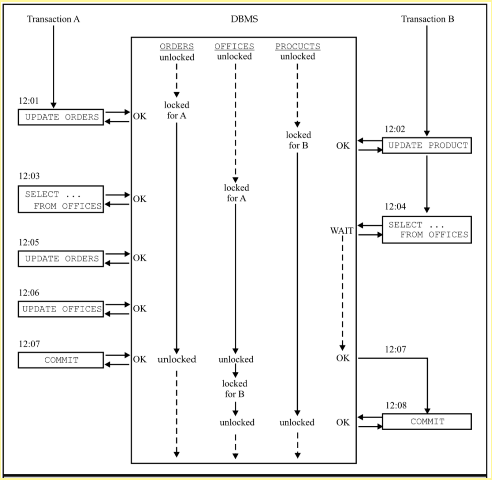
During a transaction, the user will see a completely consistent view of the database. The user will never see the uncommitted changes of other users, and even committed changes made by others will not affect data seen by the user in mid-transaction.

If two transactions, A and B, are executing concurrently, the DBMS ensures that the results will be the same as they would be if either (a) Transaction A were executed first, followed by Transaction B, or (b) Transaction B were executed first, followed by Transaction A.

This concept is known as the `serializability` of transactions.

Structured Query Language

Locking



Structured Query Language

Locking Levels

Locking can be implemented at various levels of the database.

- Database Level
  - lock the entire database for each transaction
  - only one transaction at a time
  - slow performance
  - suitable when modifying the structure of the database or scanning many large tables.

Locking Levels

Table Level

In a `table-level` locking

- DBMS locks *only* tables accessed by a transaction.
- other transactions can access other tables
- slow performance when many users must share access to the same tables

Locking Levels

Page Level

- DBMS locks individual blocks of data (pages) from the disk as they are accessed by a transaction.
- other transactions cannot access locked pages
- other transactions can access other pages of data.
- Page sizes of 2KB, 4KB, and 16KB are commonly used.
- two transactions can proceed in parallel if they access rows in different pages.

Locking Levels

Row Level

- allows two concurrent transactions that access two different rows of a table to proceed in parallel
- provides a high degree of parallel transaction execution.
- keeping track of locks is a much more complex task and it increases overhead.

Transactions

ACID Properties

Restating more concisely, we require following properties:

- *Atomicity.* Either all operations of the transaction are reflected properly in the database, or none are.
- *Consistency.* Execution of a transaction in isolation preserves the consistency of the database.
- *Isolation.* Even though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions T_i and T_j, it appears to T_i that either T_j finished execution before T_i started or T_j started execution after T_i finished.
- *Durability.* After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

Transactions

A Simple Model

A simple bank application consists of some accounts and a set of transactions that access and update those accounts:

- `read(X)`, which transfers the data item X from the database to a variable, also called X, in a buffer in main memory belonging to the transaction that executed the read operation.
- `write(X)`, which transfers the value in the variable X in the main-memory buffer of the transaction that executed the write to the data item X in the database.

[exa]

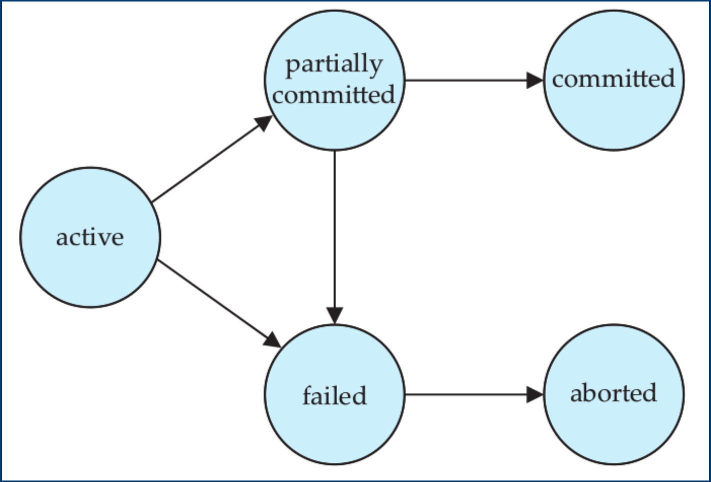
Transactions

Successful Completion

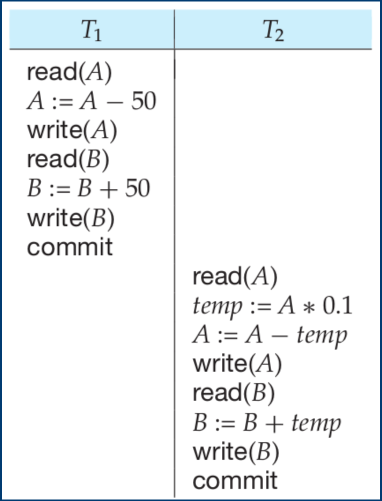
A transaction must be in one of the following states:

- *Active*, the initial state; the transaction stays in this state while it is executing.
- *Partially committed*, after the final statement has been executed.
- *Failed*, after the discovery that normal execution can no longer proceed.
- *Aborted*, after the transaction has been rolled back and the database has been restored to its state prior to the start of the transaction.
- *Committed*, after successful completion.

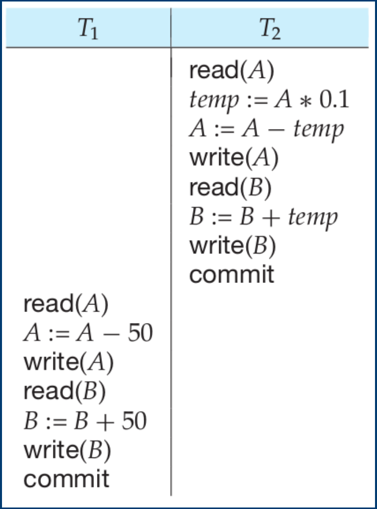
Transaction States



Schedules

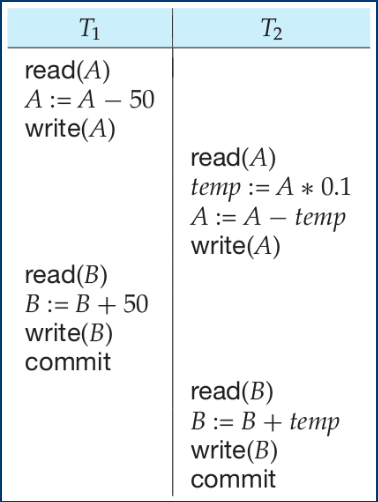


T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
read(B)	
$B := B + 50$	
write(B)	
commit	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)
	commit

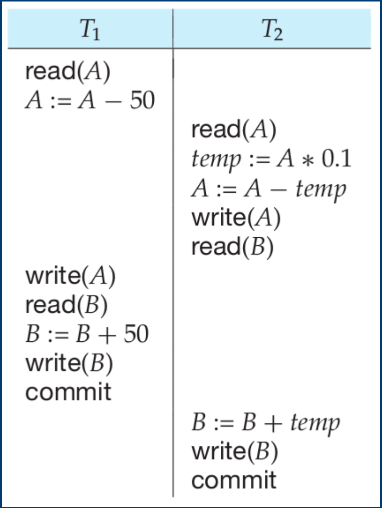


T_1	T_2
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)
	commit
read(A)	
$A := A - 50$	
write(A)	
read(B)	
$B := B + 50$	
write(B)	
commit	

Concurrency



T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
read(B)	
$B := B + 50$	
write(B)	
commit	
	read(B)
	$B := B + temp$
	write(B)
	commit



T_1	T_2
read(A)	
$A := A - 50$	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
write(A)	
read(B)	
$B := B + 50$	
write(B)	
commit	
	$B := B + temp$
	write(B)
	commit